An essential tool in Pytorch: Autograd

Rishav Das [1]

[1]*National Institute of Science Education and Research, Bhubaneswar*

Introduction[1]

torch.autograd is a vital component of PyTorch, serving as its automatic differentiation engine. It plays a crucial role in training neural networks. Here's a brief introduction:

Neural networks are essentially a collection of nested functions that operate on input data. These functions are defined by parameters (weights and biases), which are stored in tensors in PyTorch. The training of a neural network occurs in two steps: Forward Propagation and Backward Propagation.

Forward Propagation: During forward propagation, the neural network makes a prediction about the correct output by running the input data through each of its functions.

Backward Propagation: In backward propagation, the neural network adjusts its parameters in proportion to the error in its prediction. This is done by traversing backwards from the output, collecting the derivatives of the error with respect to the parameters of the functions (gradients), and optimizing the parameters using gradient descent.

torch.autograd records a graph of all operations that create the data as you execute operations, giving you a directed acyclic graph whose leaves are the input tensors and roots are the output tensors. By tracing this graph from roots to leaves, you can automatically compute the gradients using the chain rule.

This dynamic computation of gradients at runtime, even in the presence of decision branches or loops whose lengths are not known until runtime, makes PyTorch flexible and fast for building machine learning projects

References

[1] Brad Heintz. *The fundamentals of Autograd*. Apr. 2021. URL: `https://www.youtube.com/watch?v=M0fX15_-xrY&t=2s`.

[2] Elliot Waite. *Pytorch Autograd explained - in-depth tutorial*. Nov. 2018. URL: `https://www.youtube.com/watch?v=MswxJw-8PvE`.

Understanding Autograd by simple operations [2]

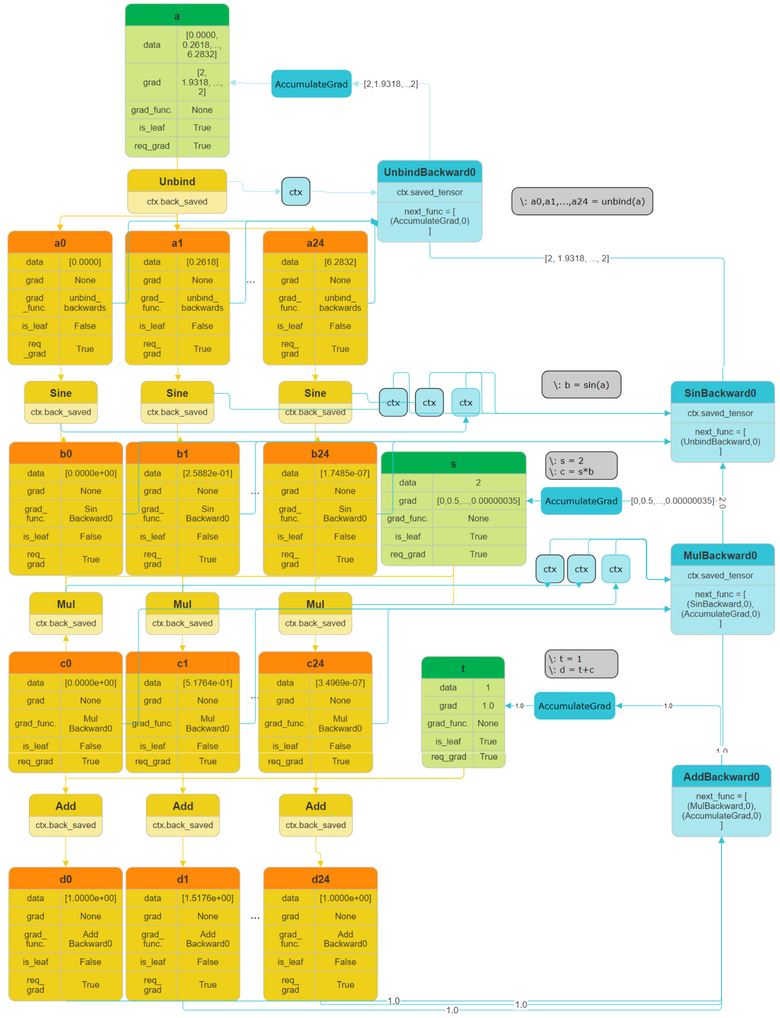

Figure 1. Example of autograd for simple opperations

Understanding Autograd in NN training

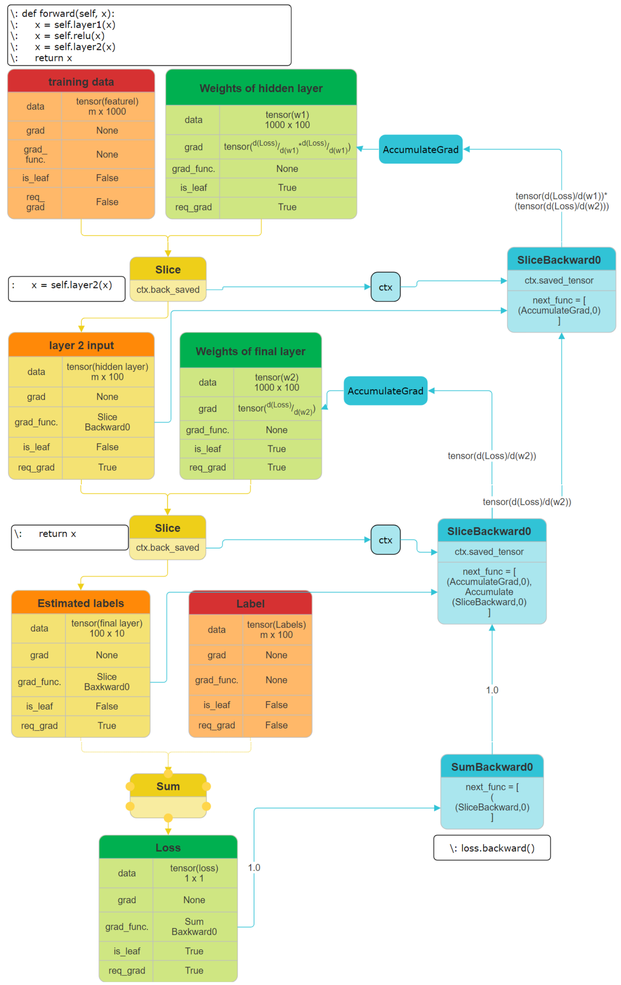

Figure 2. Autograd in action for a pass of NN

Some Important codes while using autograde in NN's :

```
optimizer = torch.optim.SGD(model.parameters(), lr=0.001)
prediction = model(some_input)
loss = (ideal_output - prediction).pow(2).sum()
loss.backward()
optimizer.step()
optimizer.zero_grad()
```